A
Dissertation
On

End To End Secured and Optimized Mobile Computing Model

Submitted in Partial Fulfillment of the Requirement
For the Award of the Degree of

Master of Technology
in
Computer Science and Engineering
by

Deepak Kumar
2K13/CSE/26

Under the Esteemed Guidance of

SH. R.K YADAV
(Assistant Professor)

2013-2016
Department of Computer Science and Engineering

DELHI TECHNOLOGICAL UNIVERSITY
JUNE 2016

DECLARATION

I hereby declare that the Major Project work entitled "**End To End Secured & Optimized Mobile Computing Model**" which is being submitted to Delhi Technological University, in partial fulfilment of requirements for the award of Degree of Master of Technology (Computer Science and Engineering) is a bonafide report of Major Project carried out by me. The material contained in the report has not been submitted to any university or institution for the award of any Degree.

DEEPAK KUMAR

University Roll no.: 2K13/CSE/26

M.Tech (Computer Science & Engineering)

Department of Computer Engineering

ACKNOWLEDGEMENT

First and foremost I would like to thank the **Lord Almighty** for showering his blessing in all endeavours.

With immense pleasure I take this opportunity to express my indebtedness gratitude to our beloved Vice chancellor **Prof. Yogesh Singh** who is enriching keen interest in academic pursuits.

I convey my sincere thanks to our Honourable HOD **Prof. O.P. Verma,** Department of CSE for his kind encouragement and motivation to complete this Project successfully.

I profoundly thank our respected Assistant Professor **Mr. R.K Yadav** Department of CSE, for his full fledged support and guidance throughout the Project.

Last but not least I render my heartiest complements to all my **Staff Members, Librarian, Family and Friends** for giving their valuable suggestions, encouragement and support for completing my project successfully.

DEEPAK KUMAR

University Roll no.: 2K13/CSE/26

M.Tech (Computer Science & Engineering)

Department of Computer Engineering

Department of Computer Engineering
DELHI TECHNOLOGICAL UNIVERSITY
Shahabad Daulatpur, Main Bawana Road,
Delhi-110042

CERTIFICATE

This is to Certify that the dissertation titled **"End To End Secured & Optimized Mobile Computing Model "** is a bonafide record of work done by **DEEPAK KUMAR, ROLL NO.: 2K13/CSE/26 at Delhi Technological University** for partial fulfilment of the requirement for the Degree of Master of Technology in Computer Science & Engineering. This project was carried out under my supervision and has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma to the best of my knowledge and belief.

Sh. R. K Yadav
Date: _____
Assistant Professor and Project Guide
Department of Computer Science and Engineering
Delhi Technological University

ABSTRACT

End-to-End Secured and Optimized Mobile Computing Model describes a Mobile Application and software technology that makes it possible to run in an area having low bandwidth maintaining the security of the data travelled through insecure channel. Web technology in conjunction with today's mobile devices and the emerging wireless technologies (e.g., digital cellular, packet radio) offer the potential for unprecedented access to data and applications by mobile workers. Yet, the limited bandwidth, high latency, high cost, and poor reliability of today's wireless wide-area networks greatly inhibits supporting such applications over wireless networks.

End to End secured and optimized model allows user to run the application on android platform to encrypt the data before it is transmitted over the wireless network and decrypt data before being shown on display structure of mobile device. It allows security of information in transmission and reception process. As data communication happens through wireless link before reaching base station from source, End to End security of data ensures data sent from the source to destination must preserve data security. In addition, it presents system for enhancing performance that reduces data volume and latency of wireless communications by performing various optimizations like compression. It makes possible to run World Wide Web applications in wide area wireless networks. This report presents the End-to-End Secured and Optimized Mobile Computing Model that makes it possible to run such handy applications efficiently in mobile networks.

Table of Contents

Table of Figures

Chapter 1

Introduction

In the present era, Mobile Computing [1] is gaining recognition as it has changed the complete landscape of our day-to-day life. The extraordinary growth evident in the area of mobile technologies creates a dynamic environment that produces diverse wireless technologies and standards, in contrast to other areas of communications marked by convergence toward uniformity. Mobile computing is an information management platform that is independent of location and time based constraints. Their mobility ensures that they are able to carry out numerous tasks at same time and perform their stated jobs. Thus, whatever be the state of the user mobile or stationary, it does not affect the working ability of the platform. We can also call it ubiquitous or pervasive computing as we have access to computer network at any location by any person all the time. With the simultaneous exponential growth of the Internet, mobile users are now seeking Internet capabilities equivalent to those provided by a fixed network. It has stimulated interest for the so-called "**nomadic computing**" which aims to provide users with access to popular desktop applications, specially suited for mobile users and basic communication services. Because of its flexibility and provision of providing ubiquitous infrastructure, there is need to provide security at every level. Thus, an impression is created that the available resources and computing power is available on the spot, whereas in reality it is far from that location. Therefore, it is important to provide security from all these threats [2]. There are different kinds of issues within security like confidentiality, integrity, availability, legitimacy and accountability that needs to be taken care individually. Some other challenges are high cost, high latency, low bandwidth and low reliability [3].

To overcome these challenges we have proposed a model that is a variation of Client/Intercept Server model that alleviates the negative characteristics of wireless link. In the proposed model, the security to the wireless transmission will be provided through the encryption of data transmitted and to optimize the transmission, compression and adaptive approach have been proposed.

1.1 Essentials of Mobile Computing

Mobile computing is basically computational task performed by users using their handsets. Since the handsets have very limited processing power and memory, these devices by themselves do not have the capability to carry out any significant and meaningful computations and can only serve as the front end for invoking the remote applications. Mobile computation, therefore, inevitably involves invocation of application running on remote servers. In other words, mobile computation is usually achieved by interaction of a front end application running on the mobile handset with the server, seamlessly, through the medium of wireless communication.

1.2 Various Mobile Computing Models [4]

1.2.1 Peer-to-Peer Model

In peer-to-peer model, the server resides at the mobile host. In this case, mobile hosts are equal partners in distributed computations. This network has emerged as an efficient system typically used for sharing files containing audio, video, data or any digital format files and distributing services over fixed networks. However, a peer-to-peer network has no central server. Each workstation on the network shares its files equally with the others. There is no central storage or authentication of users. They are inexpensive to set up, however, they offer almost no security. There is no central security or any way to control who shares what. Users are free to create any network share points on their computers. Usually, peer-to-peer networks are composed of collection of clients that run either Windows NT Workstation or Windows 98. Windows 3.11, Windows 95, and Windows 2000 Professional also support peer-to-peer networking.

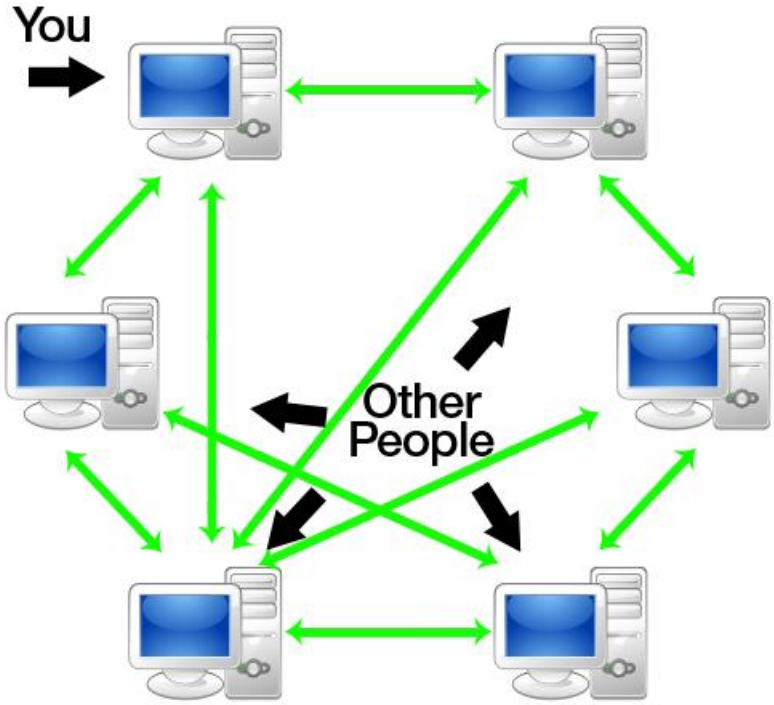


Figure 1.1: Peer to peer Model

1.2.2 Mobile-Agent Model

In the mobile-agent model [4][5][16][17] an agent first lands at an object server and then is executed to manipulate objects in the object server. If the agent finishes manipulating the objects in the object server, the agent moves to another server which has data to be manipulated. Here, agents manipulate objects only in local object servers without exchanging messages in a network. In addition to this, an agent negotiates with other agent if some agents manipulate objects in a conflicting manner. Through the negotiation, each agent

autonomously makes a decision on whether the agent continues to hold the objects or gives up the objects. After manipulating all or some of the servers, an agent makes a decision whether to commit or abort. Here, object servers may suffer from crash faults.

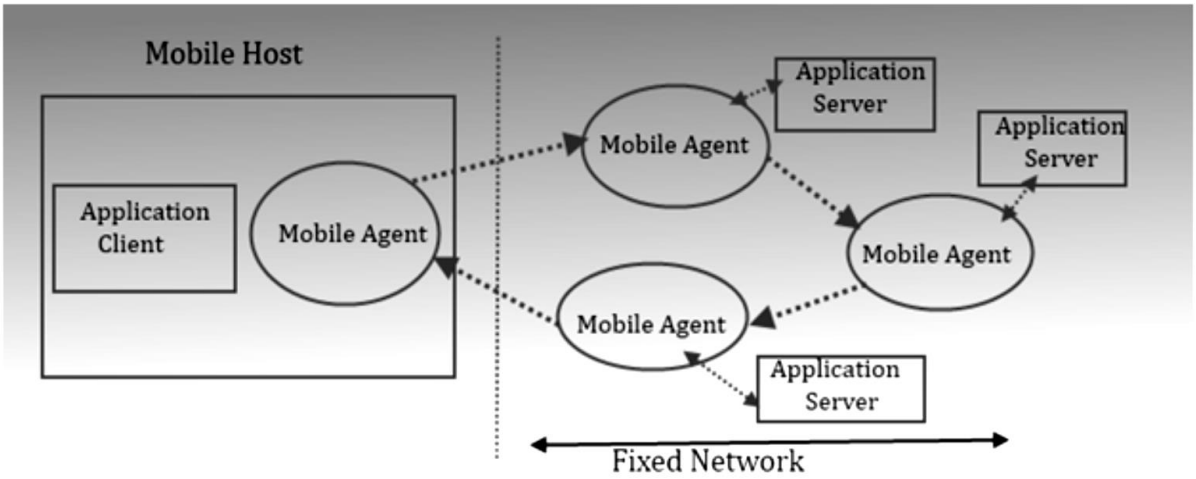


Figure 1.2: Mobile Agent Model

1.2.3 Client-Server (C/S) Model

In the client–server model, the server component provides a function or service to one or many clients, which initiate requests for such services. The C/S model requires an application or browser to be located on the mobile client and communicate directly with the web server or database server via wireless communications. At the time of accessing a specific database, the client downloads the appropriate database driver to the mobile phone and then a connection establish between client and the database server.

In case of mobile application, the limitations of traditional C/S application are as follows:

- The main limitation of traditional C/S model is that the model suffers for performance due to download and initiation of the database driver (ranges between 300-500 KB) on the client machine every time it connects to the database; that also wastes bandwidth.

- There is no way to optimize the data before the transmission to the mobile client, so receiving large data may not work or even if it works, it will take long time, which will affect the quality of the application data. So, a heavy loaded agent application may not work and regular applications will experience performance problems.

- Due to no data optimization before data transmission over network, data security cannot be provided to the wireless network. So, traditional C/S model based application undergoes security issue.

1.2.4 Client-Agent-Server (C/A/S) Model

The C/A/S architecture is a popular extension of the C/S model, containing three-tier architecture. Here, any communication goes through the mobile agent. At one extreme, agent acts as a mobile host. At another extreme, the agent is attached to a remote database or data source. Any client's request and server's response associated with this application is communicated through this service-specific agent. In this scenario, a mobile host must be associated with as many agents as the services it needs access to. Agents split the interaction between mobile clients and fixed servers into two parts, one between the client and the agent, and one between the agent and the server.

The advantages of the C/A/S model are:

- Solves the problem of initializing database driver for every query in C/S model.
- This model alleviates some of the impact of the limited bandwidth and poor reliability of wireless links by constantly maintaining the client's presence on the network via the agents.
- The agent splits the interaction between the mobile client and fixed servers into two parts, one between the client and the agent and one between the agent and the server.
- Data transmission can be optimized in the middleware so the QoS of data transmission improves with lower cost computation in the middleware or agent.A security wrapper in the middleware can provide data security over the wireless network.
- Though the client-agent-server model offers number of advantages, it fails to sustain the current computation at the mobile client during periods of disconnection. In addition, the agent can directly optimize only data transmission over the wireless link from the fixed network to the mobile client but not in the opposite direction.

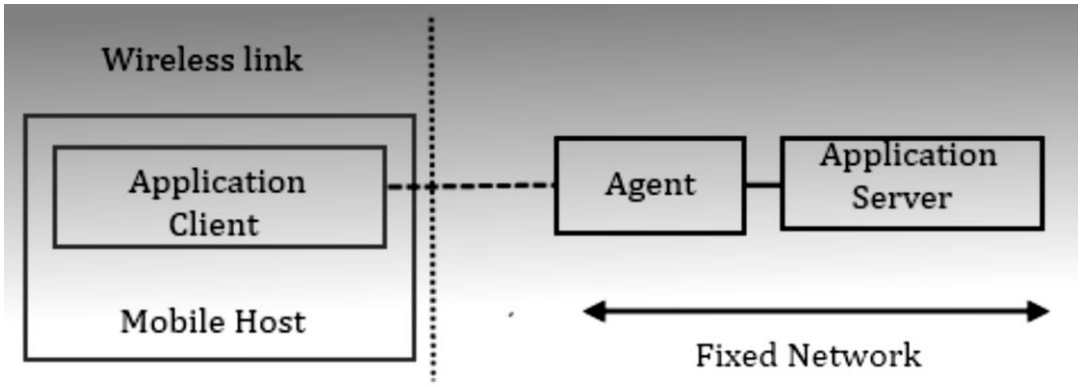


Figure 1.3: C/A/S Model

1.2.5 Client-Intercept-Server (C/I/S) Model

The C/I/S model proposes the deployment of an agent that will run at the mobile device along with an agent that will run in the server side or middleware. This client-side agent intercepts the client's requests and together with the server-side agent performs optimizations to reduce data transmission over the wireless link, improve data availability and sustain the mobile computation uninterrupted. From the point of view of the client, the client-side agent appears as the local server proxy that is co-resident with the client. Since the pair of agents is virtually inserted in the data path between the client and the server, the model is also called C/I/S instead of C/A/S.

This model provides separation of responsibly between the client-side and server-side agents which facilitate highly effective data reduction and protocol optimization. In case of database applications this model consists a client-side database agent which is specific to the agent and serve only one agent; the server side database agent will serve many agents at a time. The agent pair cooperates to intercept and control communication over the wireless link for reducing network traffic and query processing. In our research, we also investigate the C/I/S model.

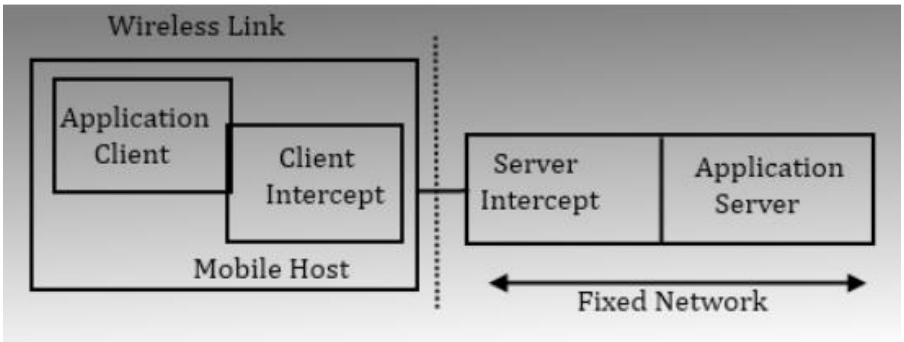


Figure 1.4: C/I/S Model

1.3 Goal, Scope and Objective of Research

Goal of research is to use the adaptive approach in extending the existing web application to have secured and optimised transmission over the mobile network. This is accomplished by enhancing the existing Client-Intercept-Server mobile computing model to End-to-End Secured and Optimised mobile computing model. The concept of using middleware in the Client intercept Server model to intercept the information transmitted from server for providing security was eliminated in the current approach. The security is provided by encoding/ decoding the data at Server or Client level through the introduction of an adaptive layer. This model is also providing secured and optimised APIs for Mobile App.

Implementation of aforementioned features has been considered as part of this thesis. End-to-end secured and optimised mobile computing model has been implemented in e-MRMS portal, an e-Governance portal for monitoring the references submitted by various Members of Legislative Assemblies to the Government for redressal. Existing portal of e-MRMS was not robust in terms of security and performance in case of mobile transmission. As part of this research, we have developed an enhanced model which caters for all these requirements.

1.4 Thesis Structure

The thesis contains five chapters. Chapter 1 describes the introduction to mobile computing. Here we have discussed about the essentials of mobile computing and various mobile computing models like peer-to-peer, client-agent-server, client-intercept-server etc. This chapter also focuses on important goal, scope and objective of study.

Chapter 2 describes client-intercept-server related work along with its usage in existing implementation and structure of existing model with its challenges. This chapter also covers the related research done so far in the field of end-to-end secured & optimized mobile computing model.

Chapter 3 describes the proposed work to overcome the challenges in transmission of data in mobile network in the existing model through the use of Adaptive Layer and encoding/decoding techniques.

Chapter 4 contains details of software used for implementing end-to-end secured and optimized computing model in e-MRMS. Adaptive Layer classes, crypto classes used to encrypt/ decrypt data at mobile node and the server end, compression classes to compress/ decompress the data at both end.

Chapter 5 describes the conclusion and future work.

Chapter 2
Related Work

2.1. Client Intercept Server(C/I/S) Model

In the C/I/S model, an agent is deployed at the mobile device along with an agent that will run in the server side or middleware. This client-side agent intercepts the client's requests and together with the server-side agent optimizes the data transmission over the wireless link, improves data availability and sustains the mobile computation uninterrupted. The client-side agent appears as the local server proxy that is co-resident with the client. Since the pair of agents is virtually inserted in the data path between the client and the server, the model is called C/I/S instead of C/A/S.

This model provides a clear distinction and separation of responsibilities between the client and the server side agents. The communication protocol between the two agents can facilitate highly effective data reduction and protocol optimization. In case of database applications, this model consists of a client-side database agent that is specific to the agent and serves only one agent and the server side database agent will serve many agents at a time. The agent pair intercepts and control communication over the wireless link for reducing network traffic and query processing. The agent pair also facilitates adaptively as the two agents can dynamically divide the workload among them based on various environmental conditions. The intercept model provides upward compatibility since it is transparent to both the client and the server. Legacy and existing applications can be executed as before since the agent pair shields them from the limitations of mobility and the wireless media. This model is more appropriate for heavy-weight clients with enough computational power and secondary storage to support the client-side agent.

Figure 2.1: Overview of the C/I/S System

2.1.1 Client Intercept Server Request and Response Overview

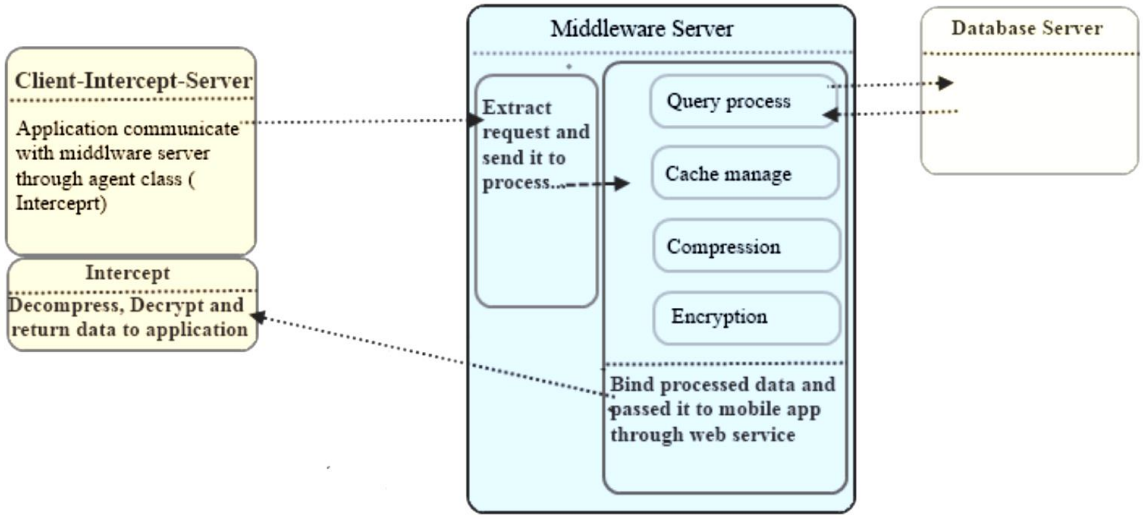


Figure 2.2: Overview of Client-Intercept-Server Request and Response

2.1.2 Middleware

Transmission of a large amount of data through a wireless network to a memory limited mobile application is a real challenge. Middleware [13][14][15] is a layer between the operating system and applications that provides a set of services

- Abstract interface to the application
- Uniform view of operating systems, networks and hardware platforms

The middleware API is called by the mobile application through an XML based web service. The web service call is a method named Get Service Data with parameters used to retrieve data.

The parameters are:

1. Database or Data Source
2. Query to execute in the Database
3. Caching Information
4. Compression Information
5. Encryption Information

Caching

In most web based application or wireless applications, caching significantly improves the performance of applications. Instead of fetching the same data repeatedly, caching can be used to store the data in a temporary memory. Every time an application fetches data, the data can come from the cache instead of recalculating it or fetching it from a remote location. In

our middleware, we use the ASP.NET application cache to cache data. The application cache provides a programmatic way to store arbitrary data in memory using key-value pairs. Using the application cache is like using application state. However, unlike application state, the data in the application cache is volatile; i.e.it is not stored in memory for the life of the application. The advantage of using the application cache is that ASP.NET manages the cache and removes items when they expire or become invalidated, or when the memory runs low.

Using a cache key the application cache determines whether an item exists in the cache or not, and if it does, to use it. If the item does not exist, the application automatically recreates the item and then places it back in the cache. The pattern of key-value pairs ensures that the cache contains the latest data from data source.

To retrieve data faster we pre-fetch data in the middleware. Every time the application creates cache data we keep a record of the cache key and the cache query. After a certain time, the application automatically checks the data and pre-fetches it in files in case the data changes.

Compression

Data compression is a common technology to represent information in a compact format. It involves encoding the information using fewer bits than its original representation. Compression is useful because it helps to reduce the consumption of resources like data space or transmission capacity. As the bandwidth of wireless network is scarce, it may be advantageous to compress the data to get the maximum out of the bandwidth. Mobile applications suffer from limited memory where data compression may help saving the memory and also improving the speed of data transfer. Compression techniques can be broadly categorized into two types :

1) Lossless Compression
   It is mainly used for spreadsheets, text and executable program compression. It is used in many applications such as the ZIP file format and in the UNIX tool gzip**.** Lossless compression is used in cases where it is important that the original data is identical to the decompressed data, or where deviations from the original data could be deleterious. Typical examples are executable programs, text documents and source code.
2) Lossy Compression
   It is mainly used for image, video, and audio compression. Lossless

Encryption

Encryption is the process of transforming the information using an algorithm to make it unreadable to anyone except to those possessing special knowledge, usually referred to as a key. The result of the process is encrypted information. The reverse process, i.e. to make the encrypted information readable again, is called decryption.

Encryption is mainly used to protect data transferred via networks, mobile telephones, wireless microphones, wireless intercom systems, Bluetooth devices and bank automatic teller machines. Encrypting data in transit helps in securing it as it is often difficult to physically secure all access to networks.

In case of enterprise databases, it is often necessary to move data over networks to other sites or to remote desktop and mobile applications. Data transmission across networks, particularly public networks, creates potential security issues. Given the importance of data security, encryption is implemented in the middleware so as to transmit data securely in the wireless network.

2.1.3 Application Programming Interface (API)

We are retrieving the data from remote data source or database. There are several mobile phones providers available in the market having their own operating system; and every operating system provides a different SDK to develop applications. Mobile application developers are facing problems due to great heterogeneity of these devices. Therefore, a common API with the flexibility to support caching, compression, and encryption will make the development of mobile application easier and faster. There is an agent API for C/I/S model which will compute the caching data, decompression and decryption and the other at the mobile application end. The two APIs developed, then, are:

1) **The Middleware API -** It provides a web service which returns data in XML format, so any mobile application can retrieve the data and use it for the application. The API has the flexibility to retrieve the data from database through middleware and make use of caching, compression, encryption or combination of these technologies.

2) **The Mobile Agent API -** It has been developed using J2ME which provides functionalities for retrieving cache data, decompressing and decrypting the data coming from the middleware API. Mobile application developers can use this API for their J2ME or Java based mobile applications.

The middleware API is a universal API providing web services and returning standard web service data so that any mobile developer can use it to retrieve data from remote databases. Also, the mobile API will help to process the remote data for the mobile application, especially developed in J2ME or Java.

2.2 Web Express for Mobile Computing

Web Express [5] is a client/intercept based system for optimizing Web browsing, reducing data volume and latency of wireless communications by intercepting the HTTP data stream and performing various optimizations including file catching, forms differencing, protocol reduction, and elimination of redundant HTTP header transmission.

An important objective of Web Express is to be able to run with any Web Browser and any Web Server without imposing any change to either.

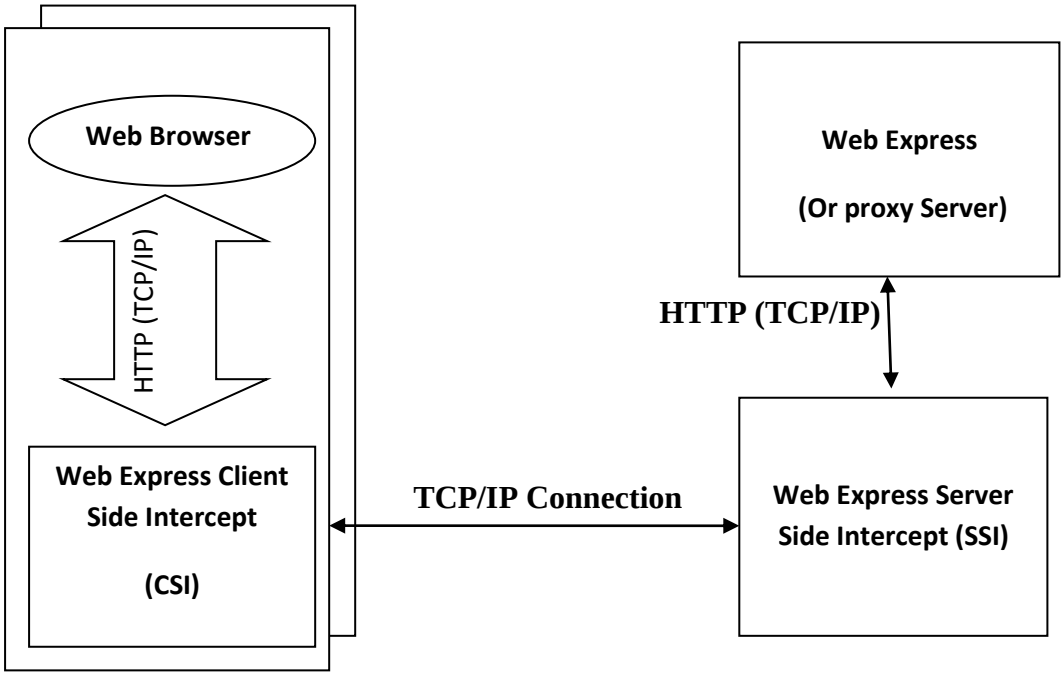


Figure 2.3: Web Express Model

In this model, two components are inserted into the data path between the Web Client and the Web Server:

1) **Client Side Intercept (CSI)** - runs in the end user client model device.
2) **Server Side Intercept (SSI)** - runs within the wire line network.

The CSI intercepts HTTP requests and, together with SSI, performs optimizations for reducing web related data transmission over the wireless link. The CSI appears as a local web proxy that is co-resident with the web browser. No other changes to the browser are required other than specifying the IP address (local) of the CSI as the browser's proxy address. The actual proxy (or socket server) address is specified as part of SSI configuration. The CSI communicates with SSI process using a reduced version of HTTP. The SSI reconstitutes the HTML data stream and forwards it to the designated web proxy server. Similarly for the response returned by web server, the CSI reconstitutes an HTML data stream received from the SSI and sends it to the Web Browser over the local TCP connection as if it came directly from the web server.

Advantages of Web Express:

- It is transparent to both Web browsers and Web (proxy) servers and therefore can be employed with any Web browser.
- The optimizations utilized by Web Express are almost totally independent of HTTP. Thus Web Express does not require to be upgraded to run with new versions of Web browsers that are available in the market place.
- The CSI/SSI protocols provide effective data reduction and protocol optimization without limiting any of the Web browser functionality or interoperability.

Web Express Optimization Methods:

- **Caching** - Both the SSI and CSI cache graphic and HTML objects. If the URL specifies an object in the CSI's cache, it is returned immediately as the browser response. The caching functions guarantee cache integrity within client-specified time interval. The SSI cache is populated by responses from the requested Web servers. If

the requested URL received from a CSI is cached in the SSI, it is returned as the response to the request.

- **Differencing** - Each new CGI request to a particular URL may result in a different response. Essential to differencing is caching a common base object on both the CSI and SSI. When a response is received, the SSI computes the difference between the base object and response and then sends the difference to the CSI. The CSI then merges the difference with its base form to create the browser response.
- **Protocol Reduction** - Each CSI connects to its SSI with a single TCP/IP connection. All requests are routed over this connection to avoid the cost of connection establishment overhead. Requests and response are multiplexed over the connection.
- **Header reduction** - Currently the HTTP protocol is stateless requiring that each request contains the browser's capabilities called access list. For a given browser, this information is same for all requests. When CSI establishes a connection with its SSI, it sends its capabilities on the first request. This information in maintained by SSI for the duration of the connection.

2.3 Client/Intercept Based System for Optimizing Wireless Access to Web Services [6]

Wireless/mobile computing is a very challenging research area due to the low data rates usually available. Moreover, wireless connections suffer bandwidth issues and are significantly less reliable than their wired counterparts and could be interrupted for various reasons (e.g., handovers).

Additionally, the cost for a wireless data connection is more. The cost per byte transmitted over the wireless interface is considerably higher as compared to wired infrastructures. Consequently, our objective is to reduce data volume to be exchanged over the wireless medium. The efforts should concentrate on the development of a model that is able to support multiple types of applications, including current and emerging TCP/IP applications and terminal emulation.

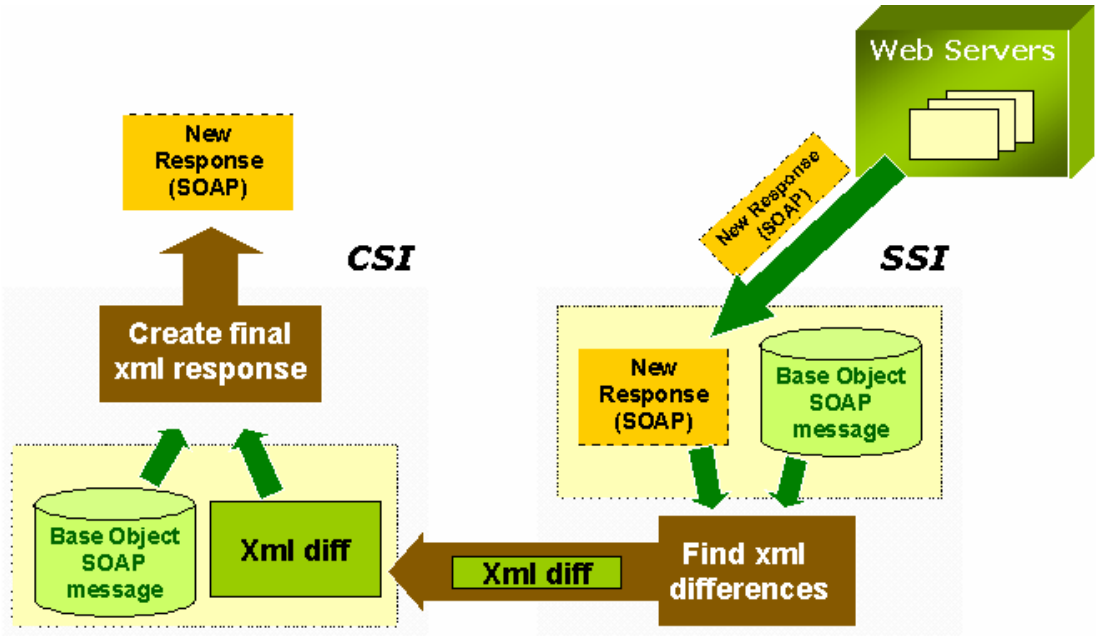


Figure 8 : Client/Intercept Based System

The basic architectural model of Web Express consists of the Client Side Intercept (CSI) running in the mobile device and the Server Side Intercept (SSI) running in the fixed network. In Web Express, the differencing optimization technique is applied on responses containing dynamic content e.g., CGI output. Dynamic responses are not stored in cache but maintained as base objects: A common base object, associated with the resource, is created in both the CSI and the SSI. Subsequent references to the resource will trigger, at the SSI, the computation of differences between its present form and the common base object. Such differences are transmitted over the radio interface. The CSI reconstitutes the referenced object and delivers it to the browser.

2.4 Challenges in existing Architecture

- Data transmission in the wireless network can be vulnerable to security attacks and, therefore, ensuring data security is an important concern in some situations.
- There is a concept of using middleware in the C/I/S model to intercept the information transmitted from server for providing security. This may result to problem when request is travelling till interceptor. Before reaching the interceptor the request travels unsecured and without optimization.
- Encryption and Decryption methodology has also not been explored in Client Intercept Server model.
- It uses techniques of cryptography method in which data decryption becomes easy if someone gets their hands on the key.
- Web Express does not mention anything for interacting through mobile applications, it only mentions about accessing the data on the mobile through web browser.

Chapter 3:
Proposed Work

3.1. Problem Statement

In the existing solution, an interceptor is deployed in transmission link both at the client and the server side. For instance when a user request a web page, the request is first intercepted by the request interceptor and then forward to server which returns the desired web page. Here the problem lies when request is travelling till interceptor. Before reaching the interceptor the request travels unsecured and without optimization.

3.2. Proposed Solution

In the proposed model, an adaptive approach has been used in extending the existing web application for secured and optimised transmission over the mobile network. This is achieved by enhancing the existing C/I/S mobile computing model to End-to-End Secured and Optimised mobile computing model. The concept of using middleware [12] in the C/I/S model to intercept the information transmitted from server for providing security is eliminated in the current approach. The security is provided by encoding/ decoding the data at Server and Client level through the introduction of an adaptive layer.
Every request from client, whether it is from desktop browser, mobile browser or mobile application, passes through this Adaptive Layer and then reaches the server. Similarly every response from server passes through the same Adaptive layer and then reaches the client.

3.3 Proposed Architecture

The figure below depicts the new architecture proposed for our solution. As shown in the figure, we have introduced an adaptive layer to identify the request type and to optimize and secure the transmission over mobile network.

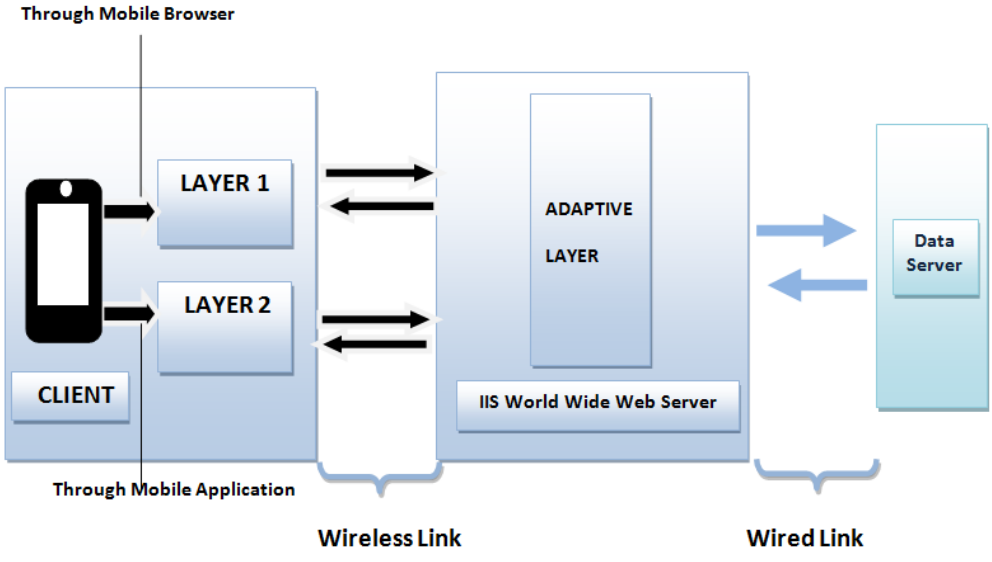


Figure 3.1: Architecture of Proposed End-End Secured & Optimized Mobile Computing Model

3.4 Flow Diagram

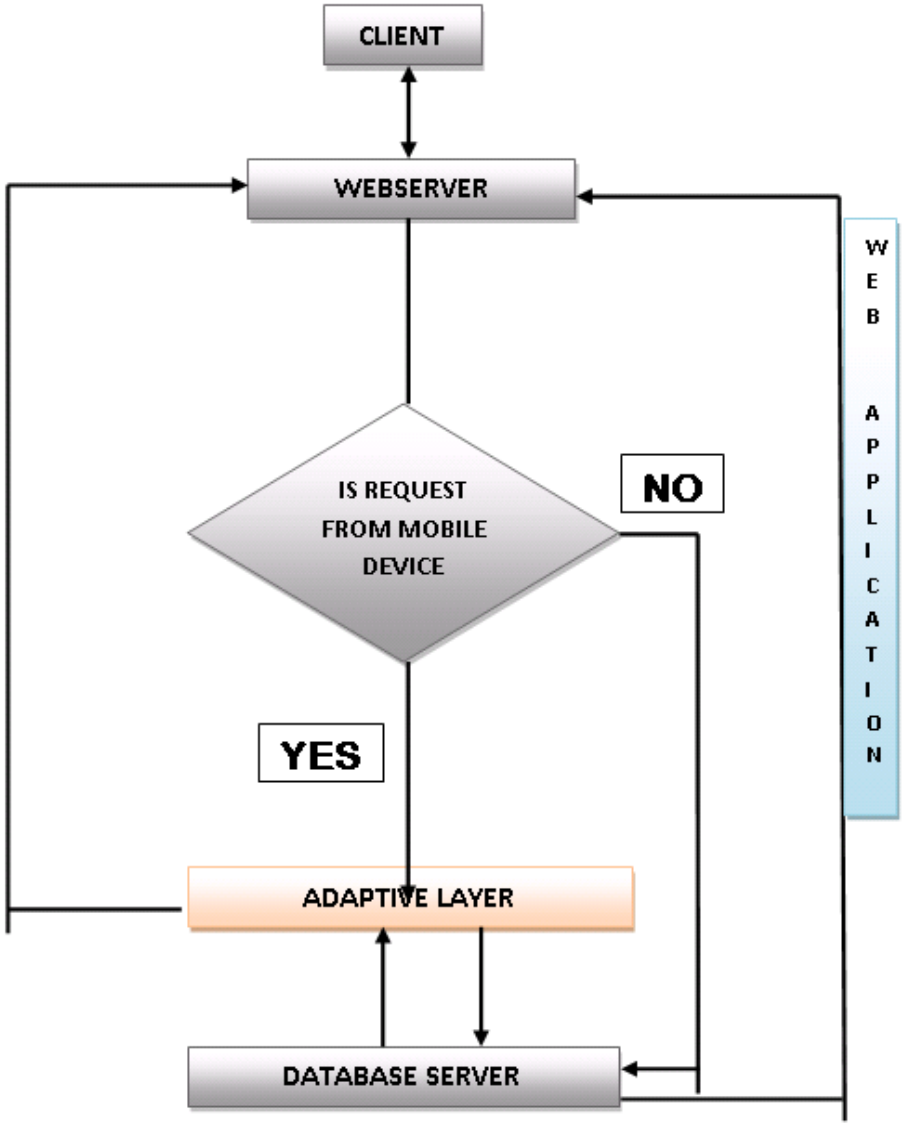


Figure 3.2: Flow Diagram of Proposed End 2 End Secured & Optimized Mobile Computing Model

3.5 Components of the Proposed Architecture

3.5.1 Client application: Client application is the end user interface which will be consumed by end users of the system. The request to the server can be from

- Desktop Browser,
- Mobile Browser or
- Mobile App.

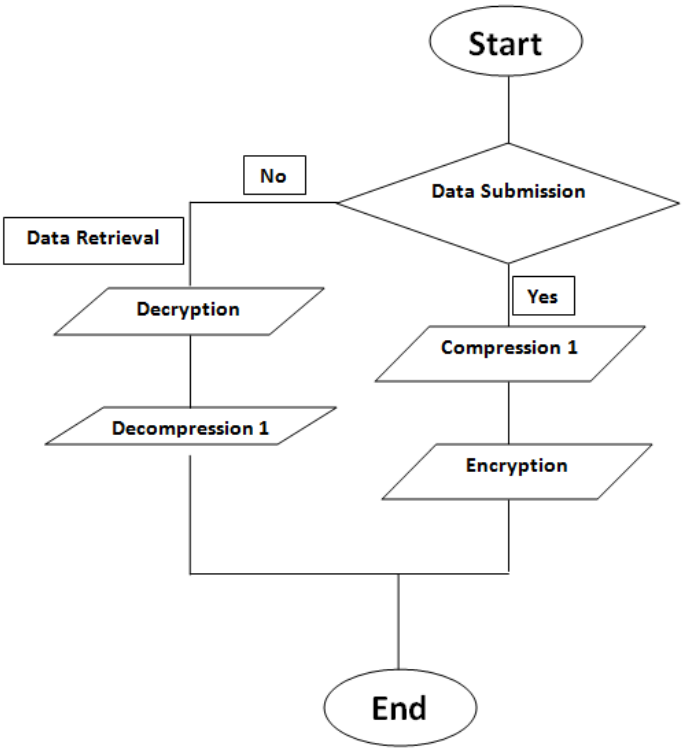


Figure 3.6: Layer 1 Architecture for Mobile Browser or Mobile App

In case of data submission request through mobile browser or App, the data to be submitted is compressed and then encrypted at the client side. The encoded data then passes through the wireless media to the Adaptive Layer.

In case of data retrieval request through mobile browser or App, the encoded data received from Adaptive Layer through wireless media is decrypted and decompressed and then displayed.

3.5.2 Adaptive Layer:

When a request reaches Adaptive Layer, it first checks the request type and takes the following action:

- **Desktop Request:** A desktop request is forwarded to the server directly without encoding the request and similarly the response is also forwarded to the client directly.

- **Mobile Request (through Mobile Browser or Mobile App):** At the server side, the encoded data received from client is decrypted and then decompressed and is sent to the database server.
  In case of data retrieval, the data received from the database server is compressed and then encrypted at the Adaptive Layer. This encoded data is then forwarded to the client through wireless media.

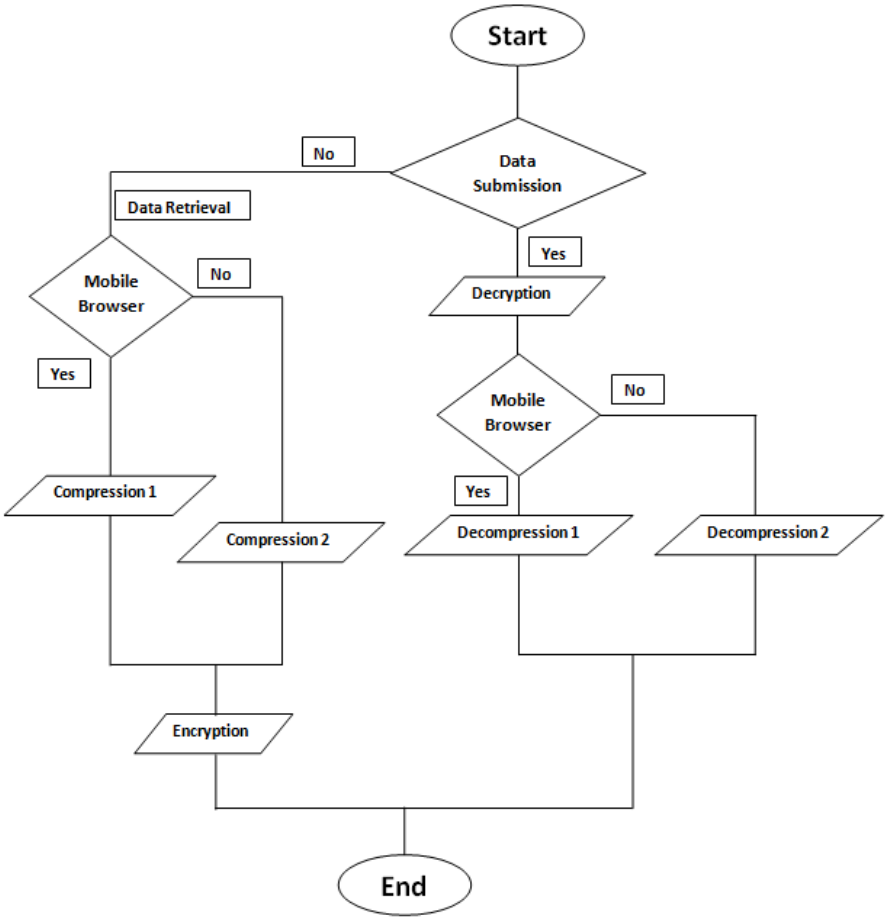


Figure 3.7: Adaptive Layer Architecture for Web Server

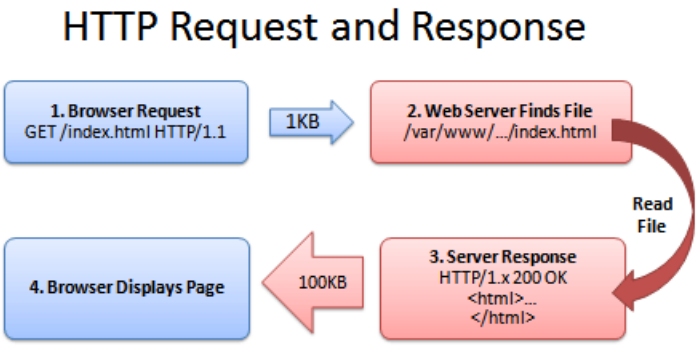
Different compression/decompression algorithm may be applied depending upon whether the request is from mobile browser or mobile app.

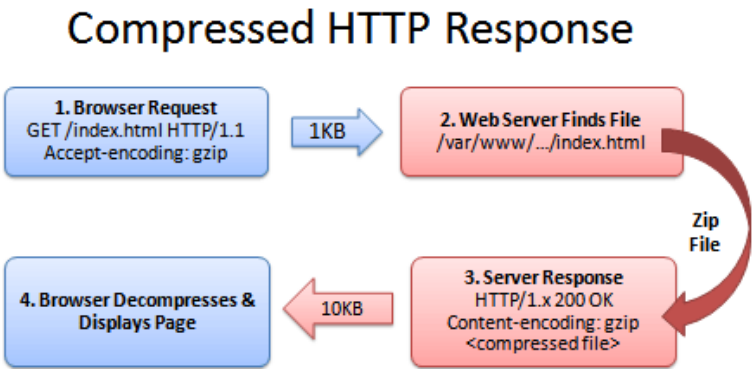
3.5.3 Web Server: Data from the client reaches the web server which after processing through the adaptive layer is sent to the database server. Similarly, the response from database server is sent to the client after processing through adaptive layer.

3.5.4 Database Server: It stores the data as per the schema.

3.6 Compression/Decompression Method Used [7][8]

In the proposed model, GZIP compression/decompression method is used. GZip is de-facto lossless compression method for compressing text data in websites. It provides an excellent trade-off between speed and compression ratio. It is the most widely used commercial compressor available.





The benefits of using GZip are:

- It is widely available in both open-source and commercial implementations.
- It provides better compression rate (40-50%) and freedom from patented algorithms
- Using gZip requires no knowledge of the document structure.
- It is built into http and web server as a standard feature

3.7 Encryption/Decryption Algorithm Used

In the proposed model, **Advanced Encryption Standard (AES) [9][10][11]** is used for encryption/decryption. The features of AES are as follows:

- Symmetric key symmetric block cipher
- 128-bit cipher, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

AES-128 where both the block and key size are 128 bits is used in the proposed model for encrypting/decrypting the data.

Chapter 4
Implementation

4.1 Software Details of Implemented System

Following are the details of system implemented

Server OS: Windows Server 2012

Database: MS SQL Server 2008

Programming Languages: C#.NET for web application and Java for mobile application

Frameworks: .NET 4.5, WCF, ASP.NET, XML

Service Hosting: IIS

4.2 Code

Following code is written at the Client Side

4.2.1 For mobile app using Java

Data Retrieval Request (Decryption + Decompression)

```
Crypto crypt=new Crypto();
CompressString cString = new CompressString();

fl.setRefid(cString.decompressBase64(crypt.decrypt(Table.getProperty("workid").toString()
, key)));

fl.setRefdtls(cString.decompressBase64(crypt.decrypt(Table.getProperty("workdetails").toS
tring(), key)));

fl.setDte_sbmsn(cString.decompressBase64(crypt.decrypt(Table.getProperty("dateofsubmis
sion").toString(), key)));

fl.setStatus(cString.decompressBase64(crypt.decrypt(Table.getProperty("status").toString(),
key)));
```

Data Submission Request (Compression + Encryption)

```
Crypto crypt = new Crypto();

CompressString cString = new CompressString();
request.addProperty("mlacode", crypt.encrypt(cString.compressBase64(params[0]),
```

```
key));
   request.addProperty("sdeptcode", crypt.encrypt(cString.compressBase64(params[1]),
key));
   request.addProperty("dtlsofissue", crypt.encrypt(cString.compressBase64(params[2]),
key));
   request.addProperty("encodedFile", params[3]);
   request.addProperty("ip", crypt.encrypt(cString.compressBase64(params[4]), key));
```

Compression/ Decompression Code

```
public class CompressString
{
   public static byte[] compress(String string) {
      try {
         ByteArrayOutputStream os = new ByteArrayOutputStream(string.length());
         GZIPOutputStream gos = new GZIPOutputStream(os);
         gos.write(string.getBytes());
         gos.close();
         byte[] compressed = os.toByteArray();
         os.close();
         return compressed;
      } catch (IOException ex) {
         return null;
      }
}

public static String compressBase64(String strToCompress)
 {
      byte[] compressed = compress(strToCompress);
      String encoded = android.util.Base64.encodeToString(compressed,
android.util.Base64.NO_WRAP);
      return encoded;
 }

public static String decompress(byte[] compressed)
 {
      try {
         final int BUFFER_SIZE = 32;
         ByteArrayInputStream is = new ByteArrayInputStream(compressed);
         GZIPInputStream gis = new GZIPInputStream(is, BUFFER_SIZE);
         StringBuilder string = new StringBuilder();
         byte[] data = new byte[BUFFER_SIZE];
         int bytesRead;
         while ((bytesRead = gis.read(data)) != -1) {
            string.append(new String(data, 0, bytesRead));
         }
         gis.close();
         is.close();
         return string.toString();
```

```
        } catch (IOException ex) {
            return null;
        }
    }

    public static String decompressBase64(String strEncoded)
    {
        byte[] decoded = android.util.Base64.decode(strEncoded,
android.util.Base64.NO_WRAP);
        String decompressed = decompress(decoded);
        return decompressed;
    }
```

Encryption/ Decryption Code

```
public class Crypto {

    private final String characterEncoding = "UTF-8";
    private final String cipherTransformation = "AES/CBC/PKCS5Padding";
    private final String aesEncryptionAlgorithm = "AES";

    public  byte[] decrypt(byte[] cipherText, byte[] key, byte [] initialVector)
            throws NoSuchAlgorithmException, NoSuchPaddingException,
InvalidKeyException,
            InvalidAlgorithmParameterException, IllegalBlockSizeException,
BadPaddingException
    {
        Cipher cipher = Cipher.getInstance(cipherTransformation);
        SecretKeySpec secretKeySpecy = new SecretKeySpec(key,
aesEncryptionAlgorithm);
        IvParameterSpec ivParameterSpec = new IvParameterSpec(initialVector);
        cipher.init(Cipher.DECRYPT_MODE, secretKeySpecy, ivParameterSpec);
        cipherText = cipher.doFinal(cipherText);
        return cipherText;
    }

    public byte[] encrypt(byte[] plainText, byte[] key, byte [] initialVector)
            throws NoSuchAlgorithmException, NoSuchPaddingException,
InvalidKeyException,
            InvalidAlgorithmParameterException, IllegalBlockSizeException,
BadPaddingException
    {
        Cipher cipher = Cipher.getInstance(cipherTransformation);
        SecretKeySpec secretKeySpec = new SecretKeySpec(key,
aesEncryptionAlgorithm);
        IvParameterSpec ivParameterSpec = new IvParameterSpec(initialVector);
        cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec, ivParameterSpec);
        plainText = cipher.doFinal(plainText);
```

```
        return plainText;
    }

    private byte[] getKeyBytes(String key) throws UnsupportedEncodingException {
        byte[] keyBytes= new byte[16];
        byte[] parameterKeyBytes= key.getBytes(characterEncoding);
        System.arraycopy(parameterKeyBytes, 0, keyBytes, 0,
 Math.min(parameterKeyBytes.length,
                keyBytes.length));
        return keyBytes;
    }

    public String encrypt(String plainText, String key) {
        String enc="";
        try {
            byte[] plainTextbytes = plainText.getBytes(characterEncoding);
            byte[] keyBytes = getKeyBytes(key);
            enc = Base64.encodeToString(encrypt(plainTextbytes, keyBytes, keyBytes),
                    Base64.DEFAULT);
        }
        catch (Exception e) {
        }
        return enc;
    }

    public String decrypt(String encryptedText, String key)
            throws KeyException, GeneralSecurityException, GeneralSecurityException,
            InvalidAlgorithmParameterException, IllegalBlockSizeException,
 BadPaddingException,
            IOException {
        byte[] cipheredBytes = Base64.decode(encryptedText, Base64.DEFAULT);
        byte[] keyBytes = getKeyBytes(key);
        return new String(decrypt(cipheredBytes, keyBytes, keyBytes),
 characterEncoding);
    }
 }
```

4.2.2 For mobile browser web application using .net C # and javascript

Data Submission Request

```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage/MasterPage.master"
AutoEventWireup="true"
   CodeFile="Ientry.aspx.cs" Inherits="Ientry" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
runat="Server">
   <script src="JS/aes.js" type="text/javascript"></script>
<script type="text/javascript">
```

```
    function Ismobile() {
        if (navigator.userAgent.match(/Android|webOS|iPhone|iPad|iPod|BlackBerry/i) != null)
{
            var wdetail = document.getElementById("<%=txt_work_detail.ClientID%>").value;
            var ddl = document.getElementById("<%=ddldepartment.ClientID%>");
            var deptval = ddl.options[ddl.selectedIndex].value;

            var key = CryptoJS.enc.Utf8.parse('8080808080808080');
            var iv = CryptoJS.enc.Utf8.parse('8080808080808080');

            var enwdetail = CryptoJS.AES.encrypt(CryptoJS.enc.Utf8.parse(wdetail), key,
                {
                    keySize: 128 / 8,
                    iv: iv,
                    mode: CryptoJS.mode.CBC,
                    padding: CryptoJS.pad.Pkcs7
                });

            var endeptval = CryptoJS.AES.encrypt(CryptoJS.enc.Utf8.parse(deptval), key,
            {
                keySize: 128 / 8,
                iv: iv,
                mode: CryptoJS.mode.CBC,
                padding: CryptoJS.pad.Pkcs7
            });

            document.getElementById("<%=hddeptval.ClientID %>").value = endeptval;
            document.getElementById("<%=hdwdetails.ClientID %>").value = enwdetail;
        }
        else {
            var ddl = document.getElementById("<%=ddldepartment.ClientID%>");
            var deptval = ddl.options[ddl.selectedIndex].value;

            document.getElementById("<%=hddeptval.ClientID %>").value =
ddl.options[ddl.selectedIndex].value;
            document.getElementById("<%=hdwdetails.ClientID %>").value =
document.getElementById("<%=txt_work_detail.ClientID%>").value; ;

            alert(deptval);
        }
    }

</script>
```

4.2.3 Code for Adaptive Layer at the Server Side

```
public class AdaptiveLayer
```

```
{
   public bool isamobileappreq()
   {
      bool flag = false;
      string userAgent =
HttpContext.Current.Request.ServerVariables["HTTP_USER_AGENT"];

      Regex OS = new
Regex(@"(android|bb\d+|meego).+mobile|avantgo|bada\/|blackberry|blazer|compal|elaine|fen
nec|hiptop|iemobile|ip(hone|od)|iris|kindle|lge
|maemo|midp|mmp|mobile.+firefox|netfront|opera m(ob|in)i|palm(
os)?|phone|p(ixi|re)\/|plucker|pocket|psp|series(4|6)0|symbian|treo|up\.(browser|link)|vodafone|
wap|windows ce|xda|xiino", RegexOptions.IgnoreCase | RegexOptions.Multiline);

      Regex device = new Regex(@"1207|6310|6590|3gso|4thp|50[1-6]i|770s|802s|a
wa|abac|ac(er|oo|s\-
)|ai(ko|rn)|al(av|ca|co)|amoi|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|attw|au(di|\-m|r |s
)|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\-(n|u)|c55\/|capi|ccwa|cdm\-
|cell|chtm|cldc|cmd\-|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\-s|devi|dica|dmob|do(c|p)o|ds(12|\-
d)|el(49|ai)|em(l2|ul)|er(ic|k0)|esl8|ez([4-7]0|os|wa|ze)|fetc|fly(\-|_)|g1 u|g560|gene|gf\-5|g\-
mo|go(\.w|od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-|hi(pt|ta)|hp( i|ip)|hs\-c|ht(c(\-|
|_|a|g|p|s|t)|tp)|hu(aw|tc)|i\-(20|go|ma)|i230|iac( |\-
|\/)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|jigs|kddi|keji|kgt( |\/)|klon|kpt
|kwc\-|kyo(c|k)|le(no|xi)|lg( g|\/(k|l|u)|50|54|\-[a-w])|libw|lynx|m1\-
w|m3ga|m50\/|ma(te|ui|xo)|mc(01|21|ca)|m\-
cr|me(rc|ri)|mi(o8|oa|ts)|mmef|mo(01|02|bi|de|do|t(\-| |o|v)|zz)|mt(50|p1|v )|mwbp|mywa|n10[0-
2]|n20[2-3]|n30(0|2)|n50(0|2|5)|n7(0(0|1)|10)|ne((c|m)\-
|on|tf|wf|wg|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|pdxg|pg(13|\-([1-
8]|c))|phil|pire|pl(ay|uc)|pn\-2|po(ck|rt|se)|prox|psio|pt\-g|qa\-a|qc(07|12|21|32|60|\-[2-7]|i\-
)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\/|sa(ge|ma|mm|ms|ny|va)|sc(01|h\-|oo|p\-)|sdk\/|se(c(\-
|0|1)|47|mc|nd|ri)|sgh\-|shar|sie(\-|m)|sk\-0|sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\-|v\-|v
)|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\-|tdg\-|tel(i|m)|tim\-|t\-mo|to(pl|sh)|ts(70|m\-
|m3|m5)|tx\-9|up(\.b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-3]|\-
v)|vm40|voda|vulc|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\-| )|webc|whit|wi(g
|nc|nw)|wmlb|wonu|x700|yas\-|your|zeto|zte\-", RegexOptions.IgnoreCase |
RegexOptions.Multiline);

      string device_info = string.Empty;

      if (OS.IsMatch(userAgent))
      {
         device_info = OS.Match(userAgent).Groups[0].Value;
      }
      if (device.IsMatch(userAgent.Substring(0, 4)))
      {
         device_info += device.Match(userAgent).Groups[0].Value;
      }

      if (string.IsNullOrEmpty(device_info) &&
(HttpContext.Current.Request.Headers["SOAPAction"] != null ||
```

```
HttpContext.Current.Request.ContentType.StartsWith("application/soap+xml")))
        {
            flag = true;
        }

        return flag;
    }

    public bool isamobilebrowser()
    {
        bool flag = false;
        string userAgent =
HttpContext.Current.Request.ServerVariables["HTTP_USER_AGENT"];

        Regex OS = new
Regex(@"(android|bb\d+|meego).+mobile|avantgo|bada\/|blackberry|blazer|compal|elaine|fen
nec|hiptop|iemobile|ip(hone|od)|iris|kindle|lge
|maemo|midp|mmp|mobile.+firefox|netfront|opera m(ob|in)i|palm(
os)?|phone|p(ixi|re)\/|plucker|pocket|psp|series(4|6)0|symbian|treo|up\.(browser|link)|vodafone|
wap|windows ce|xda|xiino", RegexOptions.IgnoreCase | RegexOptions.Multiline);

        Regex device = new Regex(@"1207|6310|6590|3gso|4thp|50[1-6]i|770s|802s|a
wa|abac|ac(er|oo|s\-
)|ai(ko|rn)|al(av|ca|co)|amoi|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|attw|au(di|\-m|r |s
)|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\-(n|u)|c55\/|capi|ccwa|cdm\-
|cell|chtm|cldc|cmd\-|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\-s|devi|dica|dmob|do(c|p)o|ds(12|\-
d)|el(49|ai)|em(l2|ul)|er(ic|k0)|esl8|ez([4-7]0|os|wa|ze)|fetc|fly(\-|_)|g1 u|g560|gene|gf\-5|g\-
mo|go(\.w|od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-|hi(pt|ta)|hp( i|ip)|hs\-c|ht(c(\-|
|_|a|g|p|s|t)|tp)|hu(aw|tc)|i\-(20|go|ma)|i230|iac( |\-
|\/)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|jigs|kddi|keji|kgt( |\/)|klon|kpt
|kwc\-|kyo(c|k)|le(no|xi)|lg( g|\/(k|l|u)|50|54|\-[a-w])|libw|lynx|m1\-
w|m3ga|m50\/|ma(te|ui|xo)|mc(01|21|ca)|m\-
cr|me(rc|ri)|mi(o8|oa|ts)|mmef|mo(01|02|bi|de|do|t(\-| |o|v)|zz)|mt(50|p1|v )|mwbp|mywa|n10[0-
2]|n20[2-3]|n30(0|2)|n50(0|2|5)|n7(0(0|1)|10)|ne((c|m)\-
|on|tf|wf|wg|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|pdxg|pg(13|\-([1-
8]|c))|phil|pire|pl(ay|uc)|pn\-2|po(ck|rt|se)|prox|psio|pt\-g|qa\-a|qc(07|12|21|32|60|\-[2-7]|i\-
)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\/|sa(ge|ma|mm|ms|ny|va)|sc(01|h\-|oo|p\-)|sdk\/|se(c(\-
|0|1)|47|mc|nd|ri)|sgh\-|shar|sie(\-|m)|sk\-0|sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\-|v\-|v
)|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\-|tdg\-|tel(i|m)|tim\-|t\-mo|to(pl|sh)|ts(70|m\-
|m3|m5)|tx\-9|up(\.b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-3]|\-
v)|vm40|voda|vulc|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\-| )|webc|whit|wi(g
|nc|nw)|wmlb|wonu|x700|yas\-|your|zeto|zte\-", RegexOptions.IgnoreCase |
RegexOptions.Multiline);

        string device_info = string.Empty;
        if (OS.IsMatch(userAgent))
        {
            device_info = OS.Match(userAgent).Groups[0].Value;
        }
        if (device.IsMatch(userAgent.Substring(0, 4)))
```

```
        {
            device_info += device.Match(userAgent).Groups[0].Value;
        }


        if (!string.IsNullOrEmpty(device_info))
        {
            flag = true;
        }

         return flag;
    }
}
```

4.2.4 Compression/ Decompression at the Server Side

```
public class Compression
{

    public DataTable compressdata(DataTable dt)
    {
        DataTable dt1 = new DataTable();

        for (int i = 0; i < dt.Columns.Count; i++)
        {
            dt1.Columns.Add(new DataColumn(dt.Columns[i].ColumnName, typeof(string)));
        }

        DataRow dr = null;
        for (int i = 0; i < dt.Rows.Count; i++)
        {
            dr = dt1.NewRow();
            for (int j = 0; j < dt.Columns.Count; j++)
            {
                dr[j] = ZipBase64(dt.Rows[i][j].ToString());
            }
            dt1.Rows.Add(dr);
        }
        return dt1;

    }

    private static void CopyTo(Stream src, Stream dest)
    {
        byte[] bytes = new byte[4096];
        int cnt;
```

```
    while ((cnt = src.Read(bytes, 0, bytes.Length)) != 0)
    {
        dest.Write(bytes, 0, cnt);
    }
}

public static byte[] Zip(string str)
{
    var bytes = Encoding.UTF8.GetBytes(str);

    using (var msi = new MemoryStream(bytes))
    using (var mso = new MemoryStream())
    {
        using (var gs = new GZipStream(mso, CompressionMode.Compress))
        {
            //msi.CopyTo(gs);
            CopyTo(msi, gs);
        }

        return mso.ToArray();
    }
}

public static string Unzip(byte[] bytes)
{
    using (var msi = new MemoryStream(bytes))
    using (var mso = new MemoryStream())
    {
        using (var gs = new GZipStream(msi, CompressionMode.Decompress))
        {
            //gs.CopyTo(mso);
            CopyTo(gs, mso);
        }

        return Encoding.UTF8.GetString(mso.ToArray());
    }
}

// Base64
public static string ZipBase64(string compress)
{
    var bytes = Zip(compress);
    var encoded = Convert.ToBase64String(bytes, Base64FormattingOptions.None);
    return encoded;
}

public static string UnzipBase64(string compressRequest)
{
    var bytes = Convert.FromBase64String(compressRequest);
```

```
        var unziped = Unzip(bytes);
        return unziped;
    }

}
```

4.2.5 Encryption/ Decryption (For mobile app request) at the Server Side

Crypto.cs

```
namespace SOAPEncrypt
{
    class Crypto
    {
        public RijndaelManaged GetRijndaelManaged(String secretKey)
        {
            var keyBytes = new byte[16];
            var secretKeyBytes = Encoding.UTF8.GetBytes(secretKey);
            Array.Copy(secretKeyBytes, keyBytes, Math.Min(keyBytes.Length,
secretKeyBytes.Length));
            return new RijndaelManaged
            {
                Mode = CipherMode.CBC,
                Padding = PaddingMode.PKCS7,
                KeySize = 128,
                BlockSize = 128,
                Key = keyBytes,
                IV = keyBytes
            };
        }

        public byte[] Encrypt(byte[] plainBytes, RijndaelManaged rijndaelManaged)
        {
            return rijndaelManaged.CreateEncryptor()
                .TransformFinalBlock(plainBytes, 0, plainBytes.Length);
        }

        public byte[] Decrypt(byte[] encryptedData, RijndaelManaged rijndaelManaged)
        {
            return rijndaelManaged.CreateDecryptor()
                .TransformFinalBlock(encryptedData, 0, encryptedData.Length);
        }

        /// <summary>
        /// Encrypts plaintext using AES 128bit key and a Chain Block Cipher and returns a
base64 encoded string
        /// </summary>
        /// <param name="plainText">Plain text to encrypt</param>
```

```
        /// <param name="key">Secret key</param>
        /// <returns>Base64 encoded string</returns>
        public String Encrypt(String plainText, String key)
        {
            var plainBytes = Encoding.UTF8.GetBytes(plainText);
            return Convert.ToBase64String(Encrypt(plainBytes, GetRijndaelManaged(key)));
        }

        /// <summary>
        /// Decrypts a base64 encoded string using the given key (AES 128bit key and a Chain
Block Cipher)
        /// </summary>
        /// <param name="encryptedText">Base64 Encoded String</param>
        /// <param name="key">Secret Key</param>
        /// <returns>Decrypted String</returns>
        public String Decrypt(String encryptedText, String key)
        {
            var encryptedBytes = Convert.FromBase64String(encryptedText);
            return Encoding.UTF8.GetString(Decrypt(encryptedBytes,
GetRijndaelManaged(key)));
        }
    }
}
```

Encryption.cs

```
public class Encryption
{
        public Encryption()
        {
                //
                // TODO: Add constructor logic here
                //
        }

   public DataTable encrpytdT(DataTable dt)
   {
      Crypto cr = new Crypto();

      string key2 = "MAKV2SPBNI99212";
      DataTable dt1 = new DataTable();

      for (int i = 0; i < dt.Columns.Count; i++)
      {
         dt1.Columns.Add(new DataColumn(dt.Columns[i].ColumnName, typeof(string)));
      }


      DataRow dr = null;
      for (int i = 0; i < dt.Rows.Count; i++)
      {
```

```
            dr = dt1.NewRow();
            for (int j = 0; j < dt.Columns.Count; j++)
            {
                dr[j] = cr.Encrypt(dt.Rows[i][j].ToString(), key2);
            }
            dt1.Rows.Add(dr);
        }
        return dt1;

    }

    public string decryptdata(string val)
    {
        string key = "MAKV2SPBNI99212";
        Crypto cr = new Crypto();
        string decryptval = cr.Decrypt(val, key);
        return decryptval;
    }

    public string encryptdata(string val)
    {
        string key = "MAKV2SPBNI99212";
        Crypto cr = new Crypto();
        string encryptval = cr.Encrypt(val, key);
        return encryptval;
    }
}
```

Encode.cs

```
public class Encode
{
        public Encode()
        {
                //
                // TODO: Add constructor logic here
                //
        }
    public DataTable encodedata(DataTable dt)
    {
        Compression objcomp = new Compression();
        DataTable compDT = objcomp.compressdata(dt);

        Encryption objenc = new Encryption();
        DataTable encDT = objenc.encrpytdT(compDT);

        return encDT;
    }
```

```
    public string decodedata(string val)
    {
        string key = "MAKV2SPBNI99212";
        Crypto cr = new Crypto();
        string decryptval = cr.Decrypt(val, key);

        string decompval = Compression.UnzipBase64(decryptval);
        return decompval;
    }
}
```

4.2.6 Encryption/ Decryption (For mobile browser request) at the Server Side

AESEncrytDecry.cs

```
public class AESEncrytDecry
{
        public AESEncrytDecry()
        {
                //
                // TODO: Add constructor logic here
                //
        }

    private static string DecryptStringFromBytes(byte[] cipherText, byte[] key, byte[] iv)
    {
        if (cipherText == null || cipherText.Length <= 0)
        {
            throw new ArgumentNullException("cipherText");
        }
        if (key == null || key.Length <= 0)
        {
            throw new ArgumentNullException("key");
        }
        if (iv == null || iv.Length <= 0)
        {
            throw new ArgumentNullException("key");
        }

        // Declare the string used to hold
        // the decrypted text.
        string plaintext = null;

        // Create an RijndaelManaged object
        // with the specified key and IV.
        using (var rijAlg = new RijndaelManaged())
```

```
        {
            rijAlg.Mode = CipherMode.CBC;
            rijAlg.Padding = PaddingMode.PKCS7;
            rijAlg.FeedbackSize = 128;

            rijAlg.Key = key;
            rijAlg.IV = iv;

            // Create a decrytor to perform the stream transform.
            var decryptor = rijAlg.CreateDecryptor(rijAlg.Key, rijAlg.IV);

            try
            {
                // Create the streams used for decryption.
                using (var msDecrypt = new MemoryStream(cipherText))
                {
                    using (var csDecrypt = new CryptoStream(msDecrypt, decryptor,
CryptoStreamMode.Read))
                    {

                        using (var srDecrypt = new StreamReader(csDecrypt))
                        {
                            // Read the decrypted bytes from the decrypting stream
                            // and place them in a string.
                            plaintext = srDecrypt.ReadToEnd();

                        }

                    }
                }
            }
            catch
            {
                plaintext = "keyError";
            }
        }

        return plaintext;
    }

    private static byte[] EncryptStringToBytes(string plainText, byte[] key, byte[] iv)
    {
        if (plainText == null || plainText.Length <= 0)
        {
            throw new ArgumentNullException("plainText");
        }
        if (key == null || key.Length <= 0)
        {
            throw new ArgumentNullException("key");
        }
```

```
        if (iv == null || iv.Length <= 0)
        {
            throw new ArgumentNullException("key");
        }
        byte[] encrypted;
        // Create a RijndaelManaged object
        // with the specified key and IV.
        using (var rijAlg = new RijndaelManaged())
        {
            rijAlg.Mode = CipherMode.CBC;
            rijAlg.Padding = PaddingMode.PKCS7;
            rijAlg.FeedbackSize = 128;

            rijAlg.Key = key;
            rijAlg.IV = iv;

            // Create a decrytor to perform the stream transform.
            var encryptor = rijAlg.CreateEncryptor(rijAlg.Key, rijAlg.IV);

            // Create the streams used for encryption.
            using (var msEncrypt = new MemoryStream())
            {
                using (var csEncrypt = new CryptoStream(msEncrypt, encryptor,
CryptoStreamMode.Write))
                {
                    using (var swEncrypt = new StreamWriter(csEncrypt))
                    {
                        //Write all data to the stream.
                        swEncrypt.Write(plainText);
                    }
                    encrypted = msEncrypt.ToArray();
                }
            }
        }
        // Return the encrypted bytes from the memory stream.
        return encrypted;
    }

    public static string DecryptStringAES(string cipherText)
    {
        var keybytes = Encoding.UTF8.GetBytes("8080808080808080");
        var iv = Encoding.UTF8.GetBytes("8080808080808080");

        var encrypted = Convert.FromBase64String(cipherText);
        var decriptedFromJavascript = DecryptStringFromBytes(encrypted, keybytes, iv);
        return string.Format(decriptedFromJavascript);
    }

    public static string EncryptStringAES(string texttoencrypt)
    {
```

```
        var keybytes = Encoding.UTF8.GetBytes("8080808080808080");
        var iv = Encoding.UTF8.GetBytes("8080808080808080");

        byte[] encryptedbytes = EncryptStringToBytes(texttoencrypt, keybytes, iv);
        return Convert.ToBase64String(encryptedbytes);
    }
}
```

4.2.7 Function at the Server Side called by web service through mobile app

```
public DataTable BindGridMLAStatus(string mla, string dept, string desc, string issue_id,
string mlacode)
    {

        AdaptiveLayer adl = new AdaptiveLayer();
        bool mobilereq = adl.isamobilereq();
        if (mobilereq)
        {
            Encode objencode = new Encode();
            DataTable encodeDT = objencode.encodedata(dt);
            return encodeDT;
        }
        else
        {
            return dt;
        }
    }
```

4.3 Output of Working Model

4.3.1 Entry Screen in mobile browser

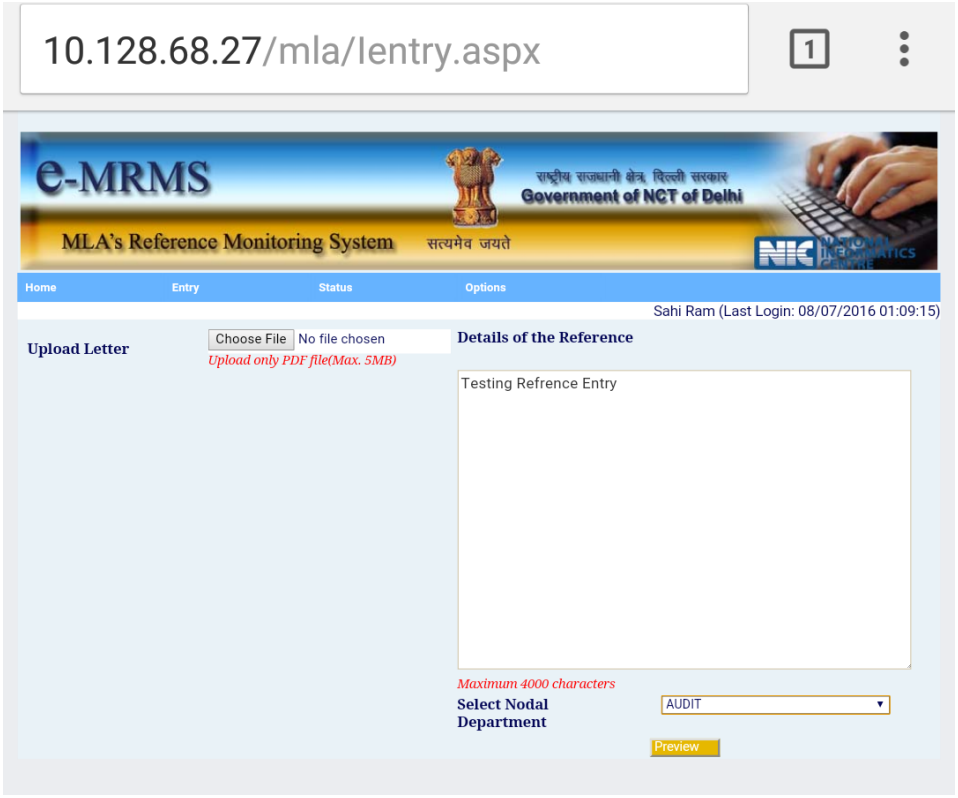


Figure 4.1 : Entry Screen in mobile browser

Encrypted Data at Client Side

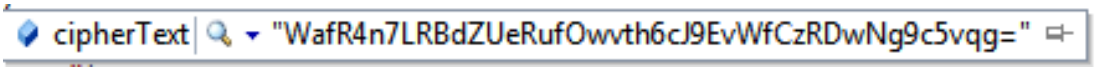


Figure 4.2 : Encrypted Data at Client Side

Decrypted Data at Server Side

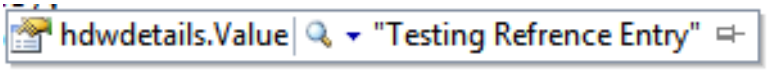


Figure 4.3 : Decrypted Data at Server Side

After data insertion in Database server

Successfully Submitted Your's Reference ID is:- 143

4.3.2 Entry Screen in mobile app

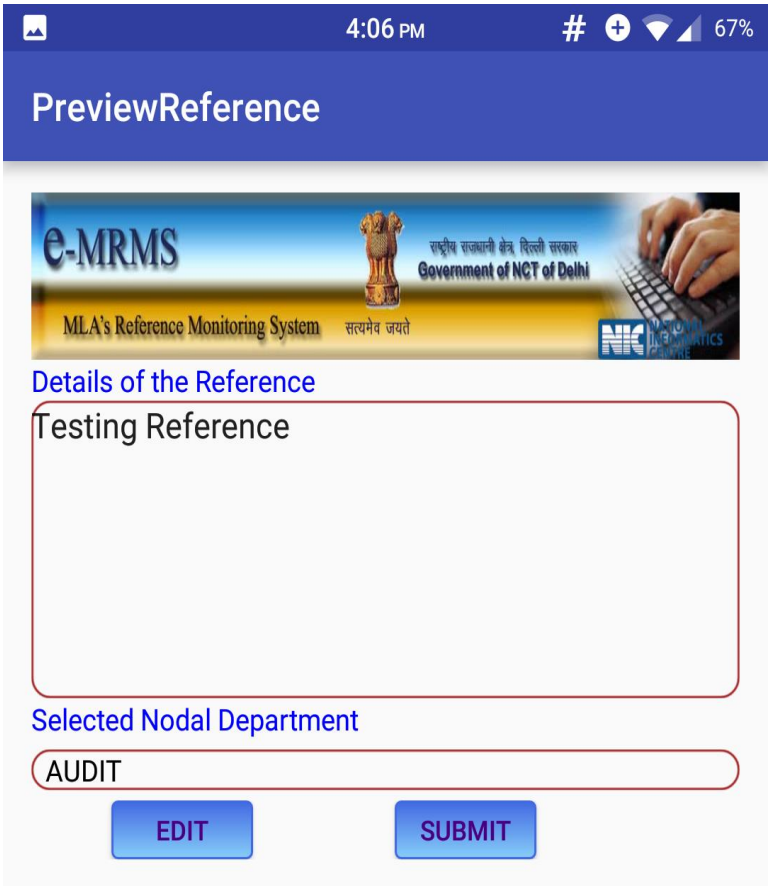


Figure 4.4: Entry Screen in mobile app

Encrypted Data at Client Side

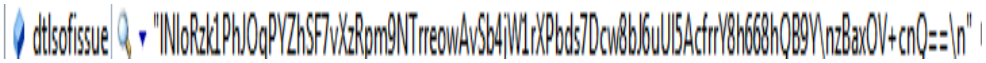


Figure 4.5: Encrypted Data at Client Side

Decrypted Data at Server Side

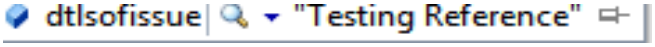
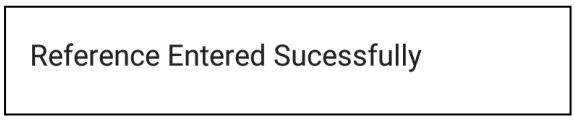


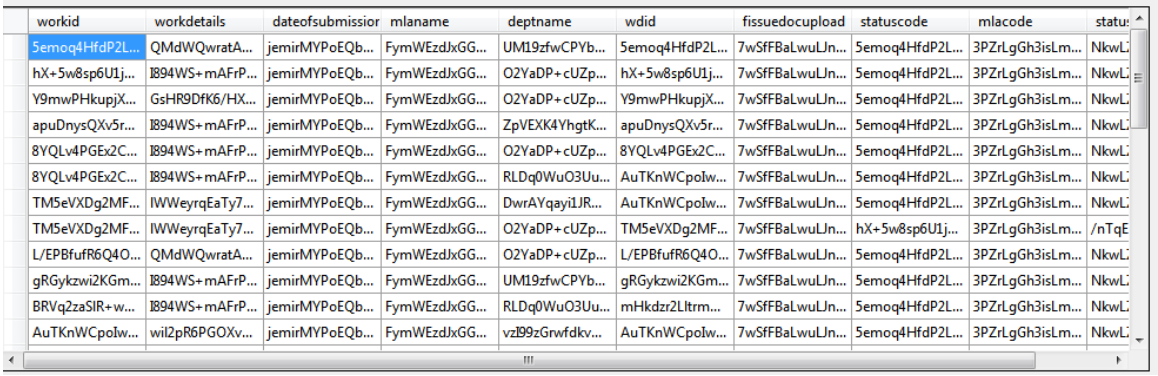
Figure 4.6: Decrypted Data at Server Side

After data insertion in Database server



4.3.3 Data Retrieval Screen in mobile app

Encrypted Data at Client Side Received from Server Side



workid	workdetails	dateofsubmissior	mlaname	deptname	wdid	fissuedocupload	statuscode	mlacode	statu
5emoq4HfdP2L...	QMdWQwratA...	jemirMYPoEQb...	FymWEzdJxGG...	UM19zfwCPYb...	5emoq4HfdP2L...	7wSfFBaLwuLJn...	5emoq4HfdP2L...	3PZrLgGh3isLm...	NkwL
hX+5w8sp6U1j...	I894WS+mAFrP...	jemirMYPoEQb...	FymWEzdJxGG...	O2YaDP+cUZp...	hX+5w8sp6U1j...	7wSfFBaLwuLJn...	5emoq4HfdP2L...	3PZrLgGh3isLm...	NkwL
Y9mwPHkupjX...	GsHR9DfK6/HX...	jemirMYPoEQb...	FymWEzdJxGG...	O2YaDP+cUZp...	Y9mwPHkupjX...	7wSfFBaLwuLJn...	5emoq4HfdP2L...	3PZrLgGh3isLm...	NkwL
apuDnysQXv5r...	I894WS+mAFrP...	jemirMYPoEQb...	FymWEzdJxGG...	ZpVEXK4YhgtK...	apuDnysQXv5r...	7wSfFBaLwuLJn...	5emoq4HfdP2L...	3PZrLgGh3isLm...	NkwL
8YQLv4PGEx2C...	I894WS+mAFrP...	jemirMYPoEQb...	FymWEzdJxGG...	O2YaDP+cUZp...	8YQLv4PGEx2C...	7wSfFBaLwuLJn...	5emoq4HfdP2L...	3PZrLgGh3isLm...	NkwL
8YQLv4PGEx2C...	I894WS+mAFrP...	jemirMYPoEQb...	FymWEzdJxGG...	RLDq0WuO3Uu...	AuTKnWCpoIw...	7wSfFBaLwuLJn...	5emoq4HfdP2L...	3PZrLgGh3isLm...	NkwL
TM5eVXDg2MF...	IWWeyrqEaTy7...	jemirMYPoEQb...	FymWEzdJxGG...	DwrAYqayi1JR...	AuTKnWCpoIw...	7wSfFBaLwuLJn...	5emoq4HfdP2L...	3PZrLgGh3isLm...	NkwL
TM5eVXDg2MF...	IWWeyrqEaTy7...	jemirMYPoEQb...	FymWEzdJxGG...	O2YaDP+cUZp...	TM5eVXDg2MF...	7wSfFBaLwuLJn...	hX+5w8sp6U1j...	3PZrLgGh3isLm...	/nTqE
L/EPBfufR6Q4O...	QMdWQwratA...	jemirMYPoEQb...	FymWEzdJxGG...	O2YaDP+cUZp...	L/EPBfufR6Q4O...	7wSfFBaLwuLJn...	5emoq4HfdP2L...	3PZrLgGh3isLm...	NkwL
gRGykzwi2KGm...	I894WS+mAFrP...	jemirMYPoEQb...	FymWEzdJxGG...	UM19zfwCPYb...	gRGykzwi2KGm...	7wSfFBaLwuLJn...	5emoq4HfdP2L...	3PZrLgGh3isLm...	NkwL
BRVq2zaSIR+w...	I894WS+mAFrP...	jemirMYPoEQb...	FymWEzdJxGG...	RLDq0WuO3Uu...	mHkdzr2LItrm...	7wSfFBaLwuLJn...	5emoq4HfdP2L...	3PZrLgGh3isLm...	NkwL
AuTKnWCpoIw...	wil2pR6PGOXv...	jemirMYPoEQb...	FymWEzdJxGG...	vzI99zGrwfdkv...	AuTKnWCpoIw...	7wSfFBaLwuLJn...	5emoq4HfdP2L...	3PZrLgGh3isLm...	NkwL

Figure 4.7: Encrypted Data at Client Side Received from Server Side for Data Retrieval

Decrypted Data at Client Side

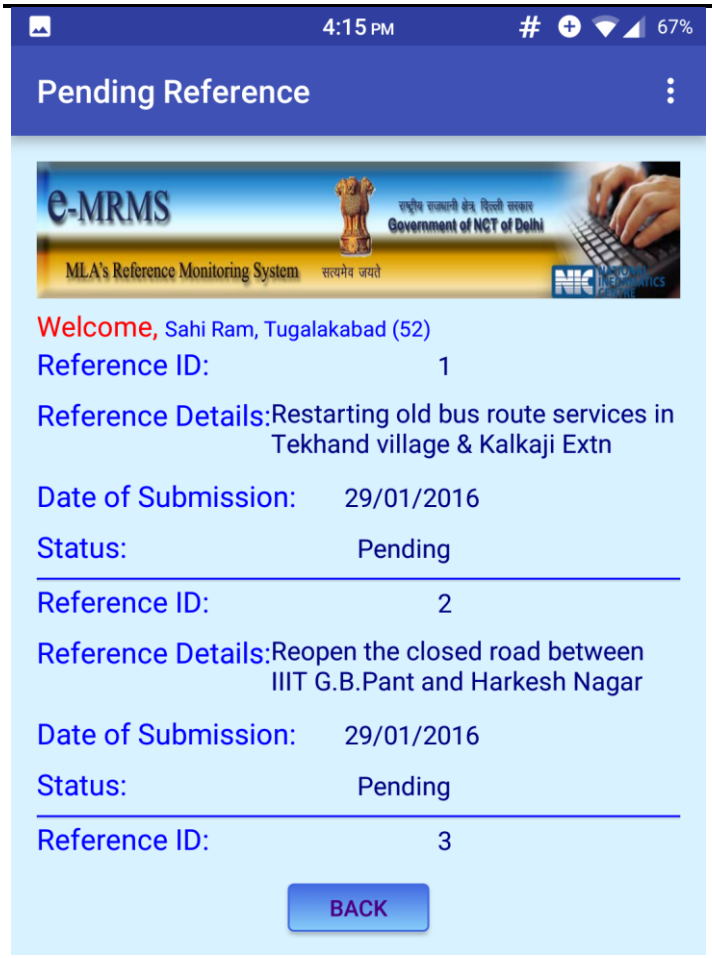


Figure 4.8: Decrypted Data at Client Side

Chapter 5
Conclusion and Future Work

An End-to-End Secured and Optimised Mobile Computing Model for a web application has been developed for extending the secured data transmission over the mobile device which overcomes the shortcomings of existing C/I/S Model. The concept of using middleware in the C/I/S model to intercept the information transmitted from server for providing security is eliminated in the proposed model. The security is provided by encoding/ decoding the data at Server and Client level through the introduction of an adaptive layer. Proposed model has features of high performance and security.

As part of future work we tend to extend this model for all the wireless media. Encryption and compression of data that is being transmitted will be explored further. There are many existing algorithms for data encryption and compression. Analysis of these encryption and compression algorithms, feasibility to implement it in End-to-End Secured and Optimised Mobile Computing Model and any enhancements to it will be part of future work.

APPENDIX A

Abbreviations

C/S - Client-Server

C/A/S – Client Agent Server

C/I/S – Client Intercept Server

API – Application Program Interface

J2ME - Java 2 Platform, Micro Edition

SDK - Software Development Kit

HTTP - Hypertext Transfer Protocol

CSI – Client Side Intercept

SSI – Server Side Intercept

HTML - HyperText Markup Language

TCP - Transmission Control Protocol

CGI - Common Gateway Interface

AES - Advanced Encryption Standard

WCF - Windows Communication Foundation

XML - Extensible Markup Language

IIS – Internet Information Services

REFERENCES

[1] Mobile Computing and Databases- A Survey: *IEEE transactions on knowledge and data engineering, vol. 11, no. 1, january/february 1999*

[2] Wireless Network Security: Vulnerabilities, Threats and Countermeasures : *International Journal of Multimedia and Ubiquitous Engineering Vol. 3, No. 3, July, 2008*

[3] Challenges of Mobile Computing: An Overview : *International Journal of Advanced Research in Computer and Communication EngineeringVol. 2, Issue 8, August 2013*

[4] Software Models for Mobile Wireless Computing: *Summer School, Jyvaskyla, August 1998*

[5] WebExpress: A Client/Intercept Based System for Optimizing Web Browsing in a Wireless Environment: *ARTICLE in MOBILE NETWORKS AND APPLICATIONS · JANUARY 1998*

[6] A Client/Intercept Based System for Optimizing Wireless Access to Web Services: *University of Athens, Department of Informatics and Telecommunications Panepistimioupolis, Ilissia, Athens 15784, Greece {grad0553, gsot, shadj}@di.uoa.gr*

[7] XML Compression Techniques: *A Survey, Department of Computer Science, University of Iowa, USA*

[8] Comparative Research of XML Compression Technologies : *DOI: 1109/ICYCS.2008.203 · Source: DBLP*

[9] Efficient Implementation of AES: *Volume 3, Issue 7, July 2013*

[10] ADVANCED ENCRYPTION STANDARD**:** *RIVIER ACADEMIC JOURNAL, VOLUME 6, NUMBER 2, FALL 2010*

[11] File Encryption, Decryption Using AES Algorithm in Android Phone**:** *Volume 5, Issue 5, May 2015*

[12] Middleware for mobile computing: Awareness vs. transparency **:** *Capra L, Emmerich W, Mascolo C*

[13] **:** A datasharing middleware for mobile computing : *Mascolo C, Capra N, Zachariadis S, and Emmerich W, Wireless Personal Communications, vol. 21, no. 1, pp. 77–103, 2002.*

[14] Context-aware reflective middleware system for mobile applications **:** *Capra L, Emmerich W, and Mascolo C***, v***ol. 29, no. 10, pp. 929–945, October 2003, doi:10.1109/TSE.2003.1237173.*

[15] Mobile agent middleware for mobile computing **:** *Bellavista P, Corradi A, and Stefanelli C:. Computer, vol. 34, no. 3, pp. 73–81, 2001.*

[16] Mobile agent model for distributed systems **:** *Komiya T, Ohsida H, Takizawa M:. 22nd International Conference on Distributed Computing Systems Workshops, 2002*.

[17] Mobile agents and security **:** *Greenberg M S, Byington J C, Harper D G, IEEE Communications Magazine, Vol.36, July 1998, pp. 76~85*